// CONNECTIONS:

// DS1302 CLK/SCLK --> 5

// DS1302 DAT/IO --> 4

// DS1302 RST/CE --> 2

// DS1302 VCC --> 3.3v - 5v

// DS1302 GND --> GND

#include <LiquidCrystal\_I2C.h>

#include <ThreeWire.h>

#include <RtcDS1302.h>

#include <Servo.h>

ThreeWire myWire(4,5,2); // IO, SCLK, CE

RtcDS1302<ThreeWire> Rtc(myWire);

LiquidCrystal\_I2C lcd(0x27, 16, 2);

Servo servo\_WhiteRice; //initialization of an instance of Servo Lib

Servo servo\_BrownRice;

Servo servo\_DrainArm;

Servo servo\_RiceCookerButton;

int i1, i2, i3;

int noOfRiceCup\_int, noOfWaterCup\_int,noOfWhiteRiceCup\_int;

int noOfBrownRiceCup\_int, spaceCount;

String noOfCupsOfBrownRice, noOfCupsOfWhiteRice;

String RTC\_Time, bluetoothmessage;

String bluetooth\_value = "0";

String riceType, noOfRiceCup, noOfWaterCup;

String RTC\_Hour\_Minute;

String bluetooth\_value\_timetrimmed;

String bluetooth\_value\_timeAndRicetypeTrimmed;

String bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed;

String hhmm;

#define redLED 3

#define greenLED 6

#define buzzerPin 11

#define waterLevel\_sensor 12

#define trigPin\_White 36

#define echoPin\_White 38

#define trigPin\_Brown 40

#define echoPin\_Brown 42

#define relaySignal\_DCMotor\_Pump\_RinsingSupply 22 //Assigning a relay signal pin (used in punp for rinsing) to digital pin 22

#define relaySignal\_DCMotor\_Pump\_CookingSupply 24 //Assigning a relay signal pin (used in punp for cooking) to digital pin 24

#define relaySignal\_DCMotor\_DrainArm\_Forward 26 //Assigning a relay signal pin (used in the motor for moving (FORWARD ROTATION) the drain) to digital pin 26

#define relaySignal\_DCMotor\_LidOpening\_Forward 32 //Assigning a relay signal pin (used in the motor for moving (FORWARD DIRECTION) the lid) to digital pin 28

#define limitSwitch\_Servo\_ButtonPress 30 //Assigning the limit switch's NO pin to digital pin 30

#define relaySignal\_DCMotor\_DrainArm\_Reverse 28 //Assigning a relay signal pin (used in the motor for moving (REVERSE ROTATION) the drain) to digital pin 32

#define relaySignal\_DCMotor\_LidOpening\_Reverse 34 //Assigning a relay signal pin (used in the motor for moving (REVERSE DIRECTION) the lid) to digital pin 34

bool executed = true;

bool executedDispensing = false;

bool oneTime = false;

void setup ()

{

// Codes For Time

Serial.begin(9600);

//Serial.print("compiled: ");

// Serial.print(\_\_DATE\_\_);

//Serial.println(\_\_TIME\_\_);

Rtc.Begin();

RtcDateTime compiled = RtcDateTime(\_\_DATE\_\_, \_\_TIME\_\_);

printDateTime(compiled);

//Serial.println();

if (!Rtc.IsDateTimeValid())

{

// Common Causes:

// 1) first time you ran and the device wasn't running yet

// 2) the battery on the device is low or even missing

// Serial.println("RTC lost confidence in the DateTime!");

Rtc.SetDateTime(compiled);

}

if (Rtc.GetIsWriteProtected())

{

// Serial.println("RTC was write protected, enabling writing now");

Rtc.SetIsWriteProtected(false);

}

if (!Rtc.GetIsRunning())

{

// Serial.println("RTC was not actively running, starting now");

Rtc.SetIsRunning(true);

}

RtcDateTime now = Rtc.GetDateTime();

if (now < compiled)

{

// Serial.println("RTC is older than compile time! (Updating DateTime)");

Rtc.SetDateTime(compiled);

}

else if (now > compiled)

{

//Serial.println("RTC is newer than compile time. (this is expected)");

}

else if (now == compiled)

{

// Serial.println("RTC is the same as compile time! (not expected but all is fine)");

}

//\*Code for Time

lcd.init();

lcd.backlight();

servo\_WhiteRice.attach(7); //declaration of the Servo Signal Pin Connections

servo\_BrownRice.attach(8);

servo\_DrainArm.attach(9);

servo\_RiceCookerButton.attach(10);

servo\_WhiteRice.write(0);

servo\_BrownRice.write(0);

servo\_DrainArm.write(0);

servo\_RiceCookerButton.write(90);

delay(1000);

servo\_WhiteRice.detach(); //declaration of the Servo Signal Pin Connections

servo\_BrownRice.detach();

servo\_DrainArm.detach();

servo\_RiceCookerButton.detach();

pinMode(buzzerPin, OUTPUT);

pinMode(relaySignal\_DCMotor\_Pump\_RinsingSupply, OUTPUT);

pinMode(relaySignal\_DCMotor\_DrainArm\_Forward, OUTPUT);

pinMode(relaySignal\_DCMotor\_DrainArm\_Reverse, OUTPUT);

pinMode(relaySignal\_DCMotor\_Pump\_CookingSupply, OUTPUT);

pinMode(relaySignal\_DCMotor\_LidOpening\_Forward, OUTPUT);

pinMode(relaySignal\_DCMotor\_LidOpening\_Reverse, OUTPUT);

pinMode(limitSwitch\_Servo\_ButtonPress, INPUT);

pinMode(redLED, OUTPUT);

pinMode(greenLED, OUTPUT);

digitalWrite(relaySignal\_DCMotor\_Pump\_RinsingSupply, HIGH);

digitalWrite(relaySignal\_DCMotor\_Pump\_CookingSupply, HIGH);

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Forward, HIGH); // DEFAULT POSITION OF DRAIN ARM

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Reverse, LOW);

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Reverse, LOW); // DEFAULT POSITION OF LID

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Forward, HIGH);

delay(3000);

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Forward, HIGH); // STOPPING THE MOTOR

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Reverse, HIGH);

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Reverse, HIGH); // STOPPING THE MOTOR

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Forward, HIGH);

riceType = "";

noOfRiceCup = "";

noOfWaterCup = "";

bluetooth\_value = "";

hhmm = "";

}

void loop ()

{

//CODES FOR RTC

RtcDateTime now = Rtc.GetDateTime();

printDateTime(now);

if (!now.IsValid())

{

// Common Causes:

// 1) the battery on the device is low or even missing and the power line was disconnected

Serial.println("RTC lost confidence in the DateTime!");

}

delay(1000); // ten seconds

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

servo\_WhiteRice.attach(7); //declaration of the Servo Signal Pin Connections

servo\_BrownRice.attach(8);

servo\_DrainArm.attach(9);

servo\_RiceCookerButton.attach(10);

servo\_WhiteRice.write(0);

servo\_BrownRice.write(0);

servo\_DrainArm.write(0);

servo\_RiceCookerButton.write(90);

delay(1000);

servo\_WhiteRice.detach(); //declaration of the Servo Signal Pin Connections

servo\_BrownRice.detach();

servo\_DrainArm.detach();

servo\_RiceCookerButton.detach();

digitalWrite(relaySignal\_DCMotor\_Pump\_RinsingSupply, HIGH);

digitalWrite(relaySignal\_DCMotor\_Pump\_CookingSupply, HIGH);

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Forward, HIGH); // DEFAULT POSITION OF DRAIN ARM

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Reverse, LOW);

digitalWrite(redLED, HIGH); //indicator for bluetooth connection

digitalWrite(greenLED, LOW);

RTC\_Time = printTime(now);

int indexOfSpace = RTC\_Time.indexOf(' ');

String dateStr = "";

String timeStr = "";

for(int a = 0; a < indexOfSpace; a++)

{

dateStr = dateStr + RTC\_Time.charAt(a);

}

int b = indexOfSpace + 1;

for( b; b < RTC\_Time.length(); b++)

{

timeStr = timeStr + RTC\_Time.charAt(b);

}

String time\_Hour = "";

for(int c = 0; c < timeStr.indexOf(':'); c++)

{

time\_Hour = time\_Hour + timeStr.charAt(c);

}

String time\_Hour\_Trimmed = "";

for(int d = timeStr.indexOf(':') + 1; d < timeStr.length(); d++)

{

time\_Hour\_Trimmed = time\_Hour\_Trimmed + timeStr.charAt(d);

}

String time\_Minute = "";

for(int e = 0; e < time\_Hour\_Trimmed.indexOf(':'); e++)

{

time\_Minute = time\_Minute + time\_Hour\_Trimmed.charAt(e);

}

String time\_Second = "";

for(int f = time\_Hour\_Trimmed.indexOf(':') + 1; f < time\_Hour\_Trimmed.length(); f++)

{

time\_Second = time\_Second + time\_Hour\_Trimmed.charAt(f);

}

/\*Serial.println();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("DATE: ");

lcd.print(dateStr);\*/

int time\_hh = time\_Hour.toInt();

int time\_mm = time\_Minute.toInt();

int time\_ss = time\_Second.toInt();

RTC\_Hour\_Minute = time\_Hour+ ":" + time\_Minute;

Serial.print("bluetooth\_value: ");

Serial.println(bluetooth\_value);

Serial.print("RTC\_Hour\_Minute: ");

Serial.println(RTC\_Hour\_Minute);

Serial.print("hhmm: ");

Serial.println(hhmm);

Serial.print("riceType: ");

Serial.println(riceType);

Serial.print("noOfRiceCup: ");

Serial.println(noOfRiceCup);

Serial.print("noOfWaterCup: ");

Serial.println(noOfWaterCup);

Serial.println("");

if(time\_hh < 10)

{

time\_Hour = "0" + time\_Hour;

}

if(time\_mm < 10)

{

time\_Minute = "0" + time\_Minute;

}

if(time\_ss < 10)

{

time\_Second = "0" + time\_Second;

}

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TIME: ");

lcd.print(time\_Hour);

lcd.print(":");

lcd.print(time\_Minute);

lcd.print(":");

lcd.print(time\_Second);

if(hhmm == "")

{

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TIME: ");

lcd.print(time\_Hour);

lcd.print(":");

lcd.print(time\_Minute);

lcd.print(":");

lcd.print(time\_Second);

lcd.setCursor(0, 1);

lcd.print("SET SCHEDULE!");

}

else

{

int semicolon = hhmm.indexOf(':');

int hhmm\_length = hhmm.length();

int h\_Int, m\_Int;

String hHour = "";

String mMinute = "";

String hhmm\_forDisplay = "";

for(int a = 0; a < semicolon; a++)

{

hHour = hHour + hhmm.charAt(a);

}

for(int b = semicolon + 1; b < hhmm\_length; b++)

{

mMinute = mMinute + hhmm.charAt(b);

}

h\_Int = hHour.toInt();

m\_Int = mMinute.toInt();

if(h\_Int < 10){

hHour = "0" + hHour;

}

if(m\_Int < 10){

mMinute = "0" + mMinute;

}

hhmm\_forDisplay = hHour +":"+ mMinute;

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TIME: ");

lcd.print(time\_Hour);

lcd.print(":");

lcd.print(time\_Minute);

lcd.print(":");

lcd.print(time\_Second);

if(executed)

{

lcd.setCursor(0, 1);

lcd.print("SET SCHEDULE!");

}

else

{

lcd.setCursor(0, 1);

lcd.print("SCHEDULE: ");

lcd.print(hhmm\_forDisplay);

}

}

if(Serial.available() > 0)

{

bluetooth\_value = Serial.readString();

digitalWrite(redLED, LOW);

digitalWrite(greenLED, HIGH);

lcd.clear();

lcd.setCursor(0, 0); //displaying the time in LCD

lcd.print("BLUETOOTH");

lcd.setCursor(0, 1);

lcd.print("CONNECTED");

delay(1000);

lcd.clear();

lcd.setCursor(0, 0); //displaying the time in LCD

lcd.print("USER INPUT: ");

lcd.setCursor(0, 1);

lcd.print(bluetooth\_value);

delay(2000);

} //end of if(Serial.available() > 0 )

spaceCount = 0;

for(int i = 0; i < bluetooth\_value.length(); i++)

{

if(bluetooth\_value.charAt(i) == ' ')

{

spaceCount = spaceCount + 1;

}

}

if(spaceCount == 3)

{

executed = false;

hhmm = "";

i1 = bluetooth\_value.indexOf(' ');

for(int j = 0; j < i1; j++)

{ //Separating the time inputted (in String) by the user from the Serial Reading from bluetooth

hhmm = hhmm+""+bluetooth\_value.charAt(j);

}

int i2 = bluetooth\_value.length();

bluetooth\_value\_timetrimmed = "";

i1 = bluetooth\_value.indexOf(' ') + 1;

for(i1 ; i1 < i2; i1++)

{ //Trimming the time from the String from Serial Reading

bluetooth\_value\_timetrimmed = bluetooth\_value\_timetrimmed +""+bluetooth\_value.charAt(i1);

}

int i3 = bluetooth\_value\_timetrimmed.indexOf(' ');

riceType = "";

for(int k = 0; k < i3; k++)

{ //Separating the Rice Type (in String) selected by the user from the Serial Reading from bluetooth

riceType = riceType+""+bluetooth\_value\_timetrimmed.charAt(k);

}

int i4 = bluetooth\_value\_timetrimmed.length();

bluetooth\_value\_timeAndRicetypeTrimmed = "";

for( int l = i3 + 1 ; l < i4; l++)

{

bluetooth\_value\_timeAndRicetypeTrimmed = bluetooth\_value\_timeAndRicetypeTrimmed +""+bluetooth\_value\_timetrimmed.charAt(l); //Trimming the Rice Type from the String from Serial Reading

}

int i5 = bluetooth\_value\_timeAndRicetypeTrimmed.indexOf(' ');

noOfRiceCup = "";

for(int m = 0; m < i5; m++)

{ //Separating the No of Cups of Rice (in String) selected by the user from the Serial Reading from bluetooth

noOfRiceCup = noOfRiceCup + ""+bluetooth\_value\_timeAndRicetypeTrimmed.charAt(m);

}

noOfRiceCup\_int = noOfRiceCup.toInt();

int i6 = bluetooth\_value\_timeAndRicetypeTrimmed.indexOf(' ') + 1;

noOfWaterCup = "";

for(int n = i6; n < bluetooth\_value\_timeAndRicetypeTrimmed.length(); n++)

{ //Separating the No of Cups of Water (in String) selected by the user from the Serial Reading from bluetooth

noOfWaterCup = noOfWaterCup + ""+bluetooth\_value\_timeAndRicetypeTrimmed.charAt(n);

}

if(noOfWaterCup == "Default")

{

if(riceType == "Brown")

{

noOfWaterCup\_int = noOfBrownRiceCup\_int \* 1.5;

}

else

{

noOfWaterCup\_int = noOfWhiteRiceCup\_int;

}

}

else

{

noOfWaterCup\_int = noOfWaterCup.toInt();

}

} // if(spaceCount == 3)

else if(spaceCount == 4)

{

executed = false;

hhmm = "";

i1 = bluetooth\_value.indexOf(' ');

for(int j = 0; j < i1; j++)

{ //Separating the time inputted (in String) by the user from the Serial Reading from bluetooth

hhmm = hhmm+""+bluetooth\_value.charAt(j);

}

int i2 = bluetooth\_value.length();

bluetooth\_value\_timetrimmed = "";

i1 = bluetooth\_value.indexOf(' ') + 1;

for(i1 ; i1 < i2; i1++)

{ //Trimming the time from the String from Serial Reading

bluetooth\_value\_timetrimmed = bluetooth\_value\_timetrimmed +""+bluetooth\_value.charAt(i1);

}

int i3 = bluetooth\_value\_timetrimmed.indexOf(' ');

riceType = "";

for(int k = 0; k < i3; k++)

{ //Separating the Rice Type (in String) selected by the user from the Serial Reading from bluetooth

riceType = riceType+""+bluetooth\_value\_timetrimmed.charAt(k);

}

int i4 = bluetooth\_value\_timetrimmed.length();

bluetooth\_value\_timeAndRicetypeTrimmed = "";

for( int l = i3 + 1 ; l < i4; l++)

{

bluetooth\_value\_timeAndRicetypeTrimmed = bluetooth\_value\_timeAndRicetypeTrimmed +""+bluetooth\_value\_timetrimmed.charAt(l); //Trimming the Rice Type from the String from Serial Reading

}

int i5 = bluetooth\_value\_timeAndRicetypeTrimmed.indexOf(' ');

noOfCupsOfWhiteRice = "";

for(int m = 0; m < i5; m++)

{ //Separating the No of Cups of Rice (in String) selected by the user from the Serial Reading from bluetooth

noOfCupsOfWhiteRice = noOfCupsOfWhiteRice + ""+bluetooth\_value\_timeAndRicetypeTrimmed.charAt(m);

}

noOfWhiteRiceCup\_int = noOfCupsOfWhiteRice.toInt();

int i7 = bluetooth\_value\_timeAndRicetypeTrimmed.length();

int i6 = bluetooth\_value\_timeAndRicetypeTrimmed.indexOf(' ') + 1;

bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed = "";

for( int n = i6 ; n < i7; n++)

{

bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed = bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed +""+bluetooth\_value\_timeAndRicetypeTrimmed.charAt(n); //Trimming the Rice Type from the String from Serial Reading

}

noOfCupsOfBrownRice = "";

int i8 = bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed.indexOf(' ');

for(int o = 0; o < i8 ; o++)

{ //Separating the No of Cups of Water (in String) selected by the user from the Serial Reading from bluetooth

noOfCupsOfBrownRice = noOfCupsOfBrownRice + ""+ bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed.charAt(o);

}

Serial.print("No. of Cup(s) of White Rice : ");

Serial.println(noOfCupsOfWhiteRice);

Serial.print("No. of Cup(s) of Brown Rice : ");

Serial.println(noOfCupsOfBrownRice);

Serial.println("");

noOfBrownRiceCup\_int = noOfCupsOfBrownRice.toInt();

noOfWaterCup = "";

for(int p = i8 + 1; p < bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed.length(); p++)

{ //Separating the No of Cups of Water (in String) selected by the user from the Serial Reading from bluetooth

noOfWaterCup = noOfWaterCup + ""+bluetooth\_value\_timeRicetypeAndRiceVolumeTrimmed.charAt(p);

}

if(noOfWaterCup == "Default")

{

noOfWaterCup\_int = (noOfBrownRiceCup\_int \* 1.5) + noOfWhiteRiceCup\_int;

}

else

{

noOfWaterCup\_int = noOfWaterCup.toInt();

}

noOfRiceCup = noOfBrownRiceCup\_int + noOfWhiteRiceCup\_int;

noOfRiceCup\_int = noOfRiceCup.toInt();

} //else if(spaceCount == 4)

/\* if(executedDispensing)

{

executedDispensing = false;

// oneTime = true;

hhmm = "";

Serial.end();

}\*/

if(RTC\_Hour\_Minute == hhmm)

{ //Monitoring the time reading from the rtc if it meets the schedule set by the user via the app

delay(500);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("SCHEDULE FOR"); //displays what process is being executed by the system

lcd.setCursor(0, 1);

lcd.print("COOKING");

digitalWrite(buzzerPin, HIGH); //alerts that it is time to cook

delay(2000);

digitalWrite(buzzerPin, LOW);

//IDENTIFYING THE RICE TYPE SELECTED BY THE USER

if(riceType == "White")

{

lcd.clear();

lcd.setCursor(0, 0); //displaying the rice type selected

lcd.print("RICE TYPE: WHITE");

lcd.setCursor(0, 1);

lcd.print("DISPENSING RICE!"); //displays what process is being executed by the system

for (int q = 0; q < noOfRiceCup\_int; q++)

{

//POURING WHITE RICE

servo\_WhiteRice.attach(7);

servo\_WhiteRice.write(150); // making the rice dispenser rotate to pour a cup of rice

delay(3000);

servo\_WhiteRice.write(0);

delay(2000);

servo\_WhiteRice.detach();

}

//executedDispensing = true;

}

else if(riceType == "Brown")

{

lcd.clear();

lcd.setCursor(0, 0); //displaying the rice type selected

lcd.print("RICE TYPE: BROWN");

lcd.setCursor(0, 1);

lcd.print("DISPENSING RICE!"); //displays what process is being executed by the system

for (int o = 0; o < noOfRiceCup\_int; o++)

{

//POURING BROWN RICE

servo\_BrownRice.attach(8);

servo\_BrownRice.write(150); // making the rice dispenser rotate to pour a cup of rice

delay(3000);

servo\_BrownRice.write(0);

delay(2000);

servo\_BrownRice.detach();

}

//executedDispensing = true;

}

else if (riceType == "WhiteBrown")

{

for (int p = 0; p < noOfWhiteRiceCup\_int; p++)

{

lcd.clear();

lcd.setCursor(0, 0); //displaying the rice type selected

lcd.print("RICE TYPE: WHITE");

//POURING WHITE RICE

lcd.setCursor(0, 1);

lcd.print("DISPENSING RICE!"); //displays what process is being executed by the system

servo\_WhiteRice.attach(7);

servo\_WhiteRice.write(150); // making the rice dispenser rotate to pour a cup of rice

delay(3000);

servo\_WhiteRice.write(0);

delay(2000);

servo\_WhiteRice.detach();

}

for (int q = 0; q < noOfBrownRiceCup\_int; q++)

{

//POURING BROWN RICE

lcd.setCursor(0, 0);

lcd.print("RICE TYPE: BROWN"); // displays what process is being executed by the system

lcd.setCursor(0, 1);

lcd.print("DISPENSING RICE!"); //displays what process is being executed by the system

servo\_BrownRice.attach(8);

servo\_BrownRice.write(150); // making the rice dispenser rotate to pour a cup of rice

delay(3000);

servo\_BrownRice.write(0);

delay(2000);

servo\_BrownRice.detach();

}

//executedDispensing = true;

}

//SUPPLYING WATER FOR RINSING

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("ACTIVATING PUMP");

lcd.setCursor(0, 1);

lcd.print("FOR RINSING!"); //displays what process is being executed by the system

Serial.println("Activated Pump for Rinsing");

digitalWrite(relaySignal\_DCMotor\_Pump\_RinsingSupply, LOW); //Activating the DC Motor Pump to supply water for Rinsing

delay(10000);

digitalWrite(relaySignal\_DCMotor\_Pump\_RinsingSupply, HIGH);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("DRAINING THE");

lcd.setCursor(0, 1);

lcd.print("RINSED RICE!"); //displays what process is being executed by the system

delay(10000);

Serial.println("Finished Rinsing");

/\*

//MOVING THE DRAIN ARM

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("MOVING THE DRAIN");

lcd.setCursor(0, 1);

lcd.print("OVER RICECOOKER"); //displays what process is being executed by the system

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Forward, LOW); //Activating the DC Motor to move the Drain Arm and position the Drain over the rice cooker

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Reverse, HIGH); //

delay(5000);

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Forward, HIGH); // STOPPING THE MOTOR

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Reverse, HIGH);

//IDENTIFICATION OF THE NUMBER OF CUPS OF WATER BASED ON THE INPUTS OF THE USER (DEFAULT or CUSTOMIZED)

int delayForPump = noOfWaterCup\_int \* 7098; //7098 comes from the volume flowrate of the pump. It takes 7098 milliseconds for the pump to fill 1 cup of water

//POURING THE IDENTIFIED AMOUNT OF WATER

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("ACTIVATING THE");

lcd.setCursor(0, 1);

lcd.print("PUMP FOR COOKING");

// digitalWrite(relaySignal\_DCMotor\_Pump\_CookingSupply, LOW); // the pump will pour water to the rice cooker

delay(delayForPump);

digitalWrite(relaySignal\_DCMotor\_Pump\_CookingSupply, HIGH);

//POURING THE RINSED RICE TO THE RICE COOKER

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("OPENING THE DRAIN");

lcd.setCursor(0, 1);

lcd.print("POURING THE RICE");

servo\_DrainArm.attach(9);

servo\_DrainArm.write(90); //opening the Drain to pour the rice to the rice cooker

delay(5000);

servo\_DrainArm.write(0); // closing the drain back to its default state

delay(2000);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("CLOSING THE DRAIN");

lcd.setCursor(0, 1);

lcd.print("DEFAULT");

servo\_DrainArm.detach();

//REVERSING THE DRAIN BACK TO ITS NORMAL POSITION

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("MOVING THE DRAIN");

lcd.setCursor(0, 1);

lcd.print("DEFAULT POSITION");

delay(3000);

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Forward, HIGH); //Activating the DC Motor to move the Drain Arm and position the Drain back to the default position

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Reverse, LOW);

delay(5000);

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Forward, HIGH); // STOPPING THE MOTOR

digitalWrite(relaySignal\_DCMotor\_DrainArm\_Reverse, HIGH);

//CLOSING THE LID OF THE RICE COOKER

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("CLOSING THE LID");

lcd.setCursor(0, 1);

lcd.print("OF RICE COOKER");

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Forward, LOW); //making the motor rotate in forward to close the lid

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Reverse, HIGH);

delay(3000);

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Forward, HIGH); //stopping the motor

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Reverse, HIGH);

//PRESSING THE BUTTON OF THE RICE COOKER

// for(int r = 0; r < 10 ; r++){

// int s = r \* 15;

servo\_RiceCookerButton.attach(10);

servo\_RiceCookerButton.write(0);

delay(3000);

boolean pressed = digitalRead(limitSwitch\_Servo\_ButtonPress); //monitors if the limit switch that is connected to the rice cooker is pressed

// if(pressed == HIGH)

// { //once the limit switch connected to the rice cooker is pressed, it means that the rice cooker is turned on

// r = 10;

servo\_RiceCookerButton.write(90);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("TURNING ON THE");

lcd.setCursor(0, 1);

lcd.print("RICE COOKER");

delay(2000);

servo\_RiceCookerButton.detach();

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("RICE COOKER");

lcd.setCursor(0, 1);

lcd.print("ACTIVATED");

delay(30000);

// }

//}

//MONITORING IF THE RICE COOKER IS FINISHED COOKING AND THE

/\* for ( int t = 0; t < 1; t){

boolean pressed = digitalRead(limitSwitch\_Servo\_ButtonPress);

if(pressed == HIGH){

t = 0;

delay(3000);

lcd.clear();

}

else{

t = 1;

delay(1000);

}

}

//OPENING THE LID OF THE RICE COOKER

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("OPENING THE LID");

lcd.setCursor(0, 1);

lcd.print("OF RICE COOKER ");

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Forward, HIGH); //making the motor rotate in reverse to open the lid

digitalWrite(relaySignal\_DCMotor\_LidOpening\_Reverse, LOW);

delay(3000);

digitalWrite(buzzerPin, HIGH); //alerts that it is done cooking

delay(2000);

digitalWrite(buzzerPin, LOW);

lcd.clear();

lcd.setCursor(0, 0);

lcd.print("RICE COOKING");

lcd.setCursor(0, 1);

lcd.print("COMPLETED!!!");

delay(5000);

executed = true;

\*/

} //end of if(hhmm == RTC\_Hour\_Minute )

}//end of void loop()

//FUNCTIONS FOR TIME GENERATION

#define countof(a) (sizeof(a) / sizeof(a[0]))

void printDateTime(const RtcDateTime& dt)

{

char datestring[20];

snprintf\_P(datestring,

countof(datestring),

PSTR("%02u/%02u/%04u %02u:%02u:%02u"),

dt.Month(),

dt.Day(),

dt.Year(),

dt.Hour(),

dt.Minute(),

dt.Second() );

// Serial.println(datestring);

}

String printTime(const RtcDateTime& dt)

{

char datestring[20];

String timeString;

String hourString, minuteString, secondString, dayString, monthString, yearString;

unsigned int hh, mm, ss, mons, yyyy, dd;

snprintf\_P(datestring,

countof(datestring),

PSTR("%02u/%02u/%04u %02u:%02u:%02u"),

dt.Month(),

dt.Day(),

dt.Year(),

dt.Hour(),

dt.Minute(),

dt.Second() );

dd = dt.Day();

yyyy = dt.Year();

mons = dt.Month();

hh = dt.Hour();

mm = dt.Minute();

ss = dt.Second();

hourString = String(hh);

minuteString = String(mm);

secondString = String(ss);

dayString = String(dd);

monthString = String(mons);

yearString = String(yyyy);

timeString = monthString +"/"+ dayString + "/" + yearString+ " " + hourString +":" + minuteString +":"+ secondString;

return timeString;

}